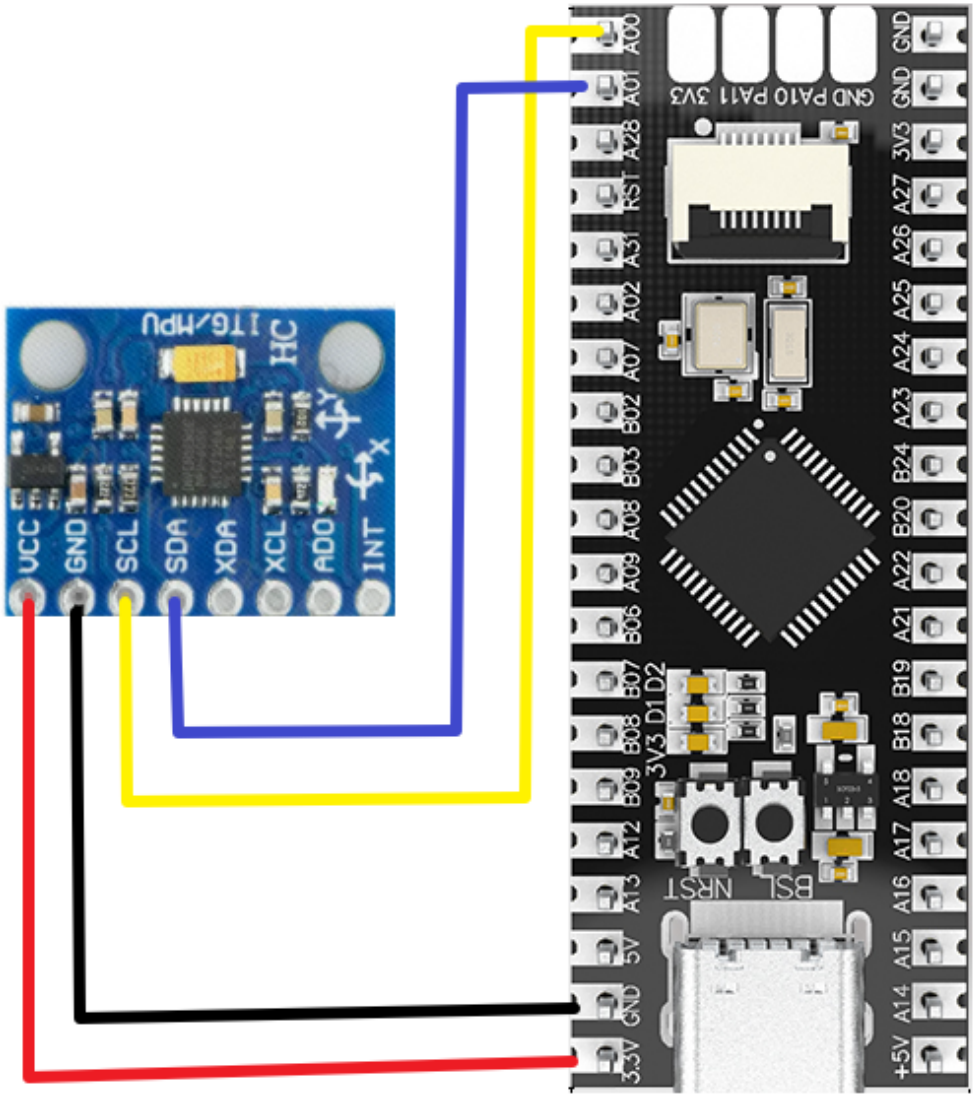
MPU6050 data acquisition

1. Learning objectives

Serial port prints MPU6050 data.

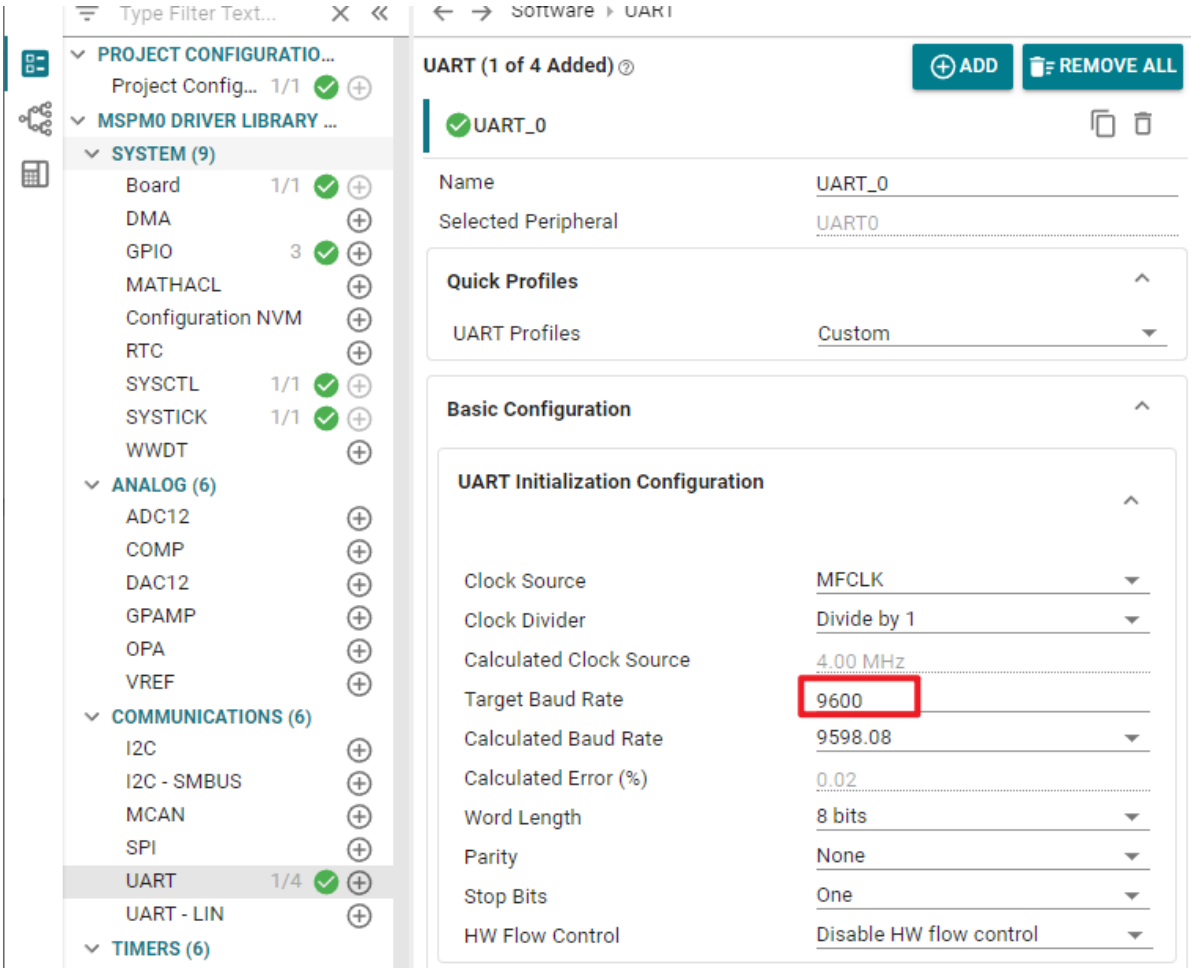
2. Hardware connection

MSPM0G3507 and MPU6050 module pin connection



MPU6050 module	MSPM0G3507
SCL	PA0
SDA	PA1
VCC	3V3
GND	GND

3. Program description



- bsp_mpu6050.h

```
#ifndef _BSP_MPU6050_H_
#define _BSP_MPU6050_H_

#include "board.h"


//Set SDA output mode
#define SDA_OUT()   {                                                   \
                        DL_GPIO_initDigitalOutput(GPIO_SDA_IOMUX);      \
                        DL_GPIO_setPins(GPIO_PORT, GPIO_SDA_PIN);       \
                        DL_GPIO_enableOutput(GPIO_PORT, GPIO_SDA_PIN); \
                    }
// Set SDA input mode
#define SDA_IN()    { DL_GPIO_initDigitalInput(GPIO_SDA_IOMUX); }

//Get the level of the SDA pin
#define SDA_GET()   ( ( ( DL_GPIO_readPins(GPIO_PORT,GPIO_SDA_PIN) & GPIO_SDA_PIN
) > 0 ) ? 1 : 0 )
//SDA and SCL output
#define SDA(x)      ( (x) ? (DL_GPIO_setPins(GPIO_PORT,GPIO_SDA_PIN)) :
(DL_GPIO_clearPins(GPIO_PORT,GPIO_SDA_PIN)) )
#define SCL(x)      ( (x) ? (DL_GPIO_setPins(GPIO_PORT,GPIO_SCL_PIN)) :
(DL_GPIO_clearPins(GPIO_PORT,GPIO_SCL_PIN)) )
```

```
//AD0 of MPU6050 is the IIC address pin. If it is grounded, the IIC address is
0x68, and if it is connected to VCC, the IIC address is 0x69


#define MPU6050_RA_SMPLRT_DIV       0x19        //Gyroscope sampling frequency
Address
#define MPU6050_RA_CONFIG           0x1A        //Set the digital low pass filter
address
#define MPU6050_RA_GYRO_CONFIG      0x1B        //Gyroscope Configuration
Registers
....
```

Defines the basic macros and register addresses for operating the MPU6050 sensor, mainly used to communicate with the MPU6050 sensor through the I2C interface.

- inv_mpu.c

```
u8 mpu_dmp_init(void)
{
    u8 res=0;


    res = mpu_init();

//    printf("res = %d\r\n",res);
    if(res==0)  //Initialize MPU6050
    {
        res=mpu_set_sensors(INV_XYZ_GYRO|INV_XYZ_ACCEL);//Set up all required
sensors
        if(res)return 1;
        res=mpu_configure_fifo(INV_XYZ_GYRO | INV_XYZ_ACCEL);//Setting up FIFO
        if(res)return 2;
        res=mpu_set_sample_rate(DEFAULT_MPU_HZ);    //Setting the Sample Rate
        if(res)return 3;
        res=dmp_load_motion_driver_firmware();      //Load dmp firmware
        if(res)return 4;

res=dmp_set_orientation(inv_orientation_matrix_to_scalar(gyro_orientation));//Set
gyroscope orientation
        if(res)return 5;
        res=dmp_enable_feature(DMP_FEATURE_6X_LP_QUAT|DMP_FEATURE_TAP|
//Setting the dmp function

DMP_FEATURE_ANDROID_ORIENT|DMP_FEATURE_SEND_RAW_ACCEL|DMP_FEATURE_SEND_CAL_GYRO|
            DMP_FEATURE_GYRO_CAL);
        if(res)return 6;
        res=dmp_set_fifo_rate(DEFAULT_MPU_HZ);  //Set the DMP output rate
(maximum 200Hz)
        if(res)return 7;
//      res=run_self_test();        //Self-examination
//      if(res)return 8;
        res=mpu_set_dmp_state(1);   //Enabling DMP
        if(res)return 9;
    }
```

```
    return 0;

}
```

The mpu_dmp_init function is defined to initialize the digital motion processor (DMP) of the MPU6050. This function goes through a series of steps to configure the MPU6050 and its DMP so that it can output the required data.

```
//q30 format, divisor when converting long to float.
#define q30  1073741824.0f
...
u8 mpu_dmp_get_data(float *pitch,float *roll,float *yaw)
{
    float q0=1.0f,q1=0.0f,q2=0.0f,q3=0.0f;
    unsigned long sensor_timestamp;
    short gyro[3], accel[3], sensors;
    unsigned char more;
    long quat[4];
    if(dmp_read_fifo(gyro, accel, quat, &sensor_timestamp, &sensors,&more))return
1;
    /* Gyro and accel data are written to the FIFO by the DMP in chip frame and
hardware units.
     * This behavior is convenient because it keeps the gyro and accel outputs of
dmp_read_fifo and mpu_read_fifo consistent.
    **/
    /*if (sensors & INV_XYZ_GYRO )
    send_packet(PACKET_TYPE_GYRO, gyro);
    if (sensors & INV_XYZ_ACCEL)
    send_packet(PACKET_TYPE_ACCEL, accel); */
    /* Unlike gyro and accel, quaternions are written to the FIFO in the body
frame, q30.
     * The orientation is set by the scalar passed to dmp_set_orientation during
initialization.
    **/
    if(sensors&INV_WXYZ_QUAT)
    {
        q0 = quat[0] / q30; //Convert q30 format to floating point number
        q1 = quat[1] / q30;
        q2 = quat[2] / q30;
        q3 = quat[3] / q30;
        //Calculated pitch/roll/heading angles
        *pitch = asin(-2 * q1 * q3 + 2 * q0* q2)* 57.3; // pitch
        *roll  = atan2(2 * q2 * q3 + 2 * q0 * q1, -2 * q1 * q1 - 2 * q2* q2 +
1)* 57.3; // roll
        *yaw   = atan2(2*(q1*q2 + q0*q3),q0*q0+q1*q1-q2*q2-q3*q3) * 57.3;
//yaw
    }else return 2;
    return 0;
}
```

The function u8 mpu_dmp_get_data(float *pitch,float *roll,float *yaw) calculates the pitch angle (pitch), roll angle (roll), and yaw angle (yaw) from the MPU data, and outputs these angles in the form of floating-point numbers to the location pointed to by the specified parameter pointer.

- empty.c

```
```

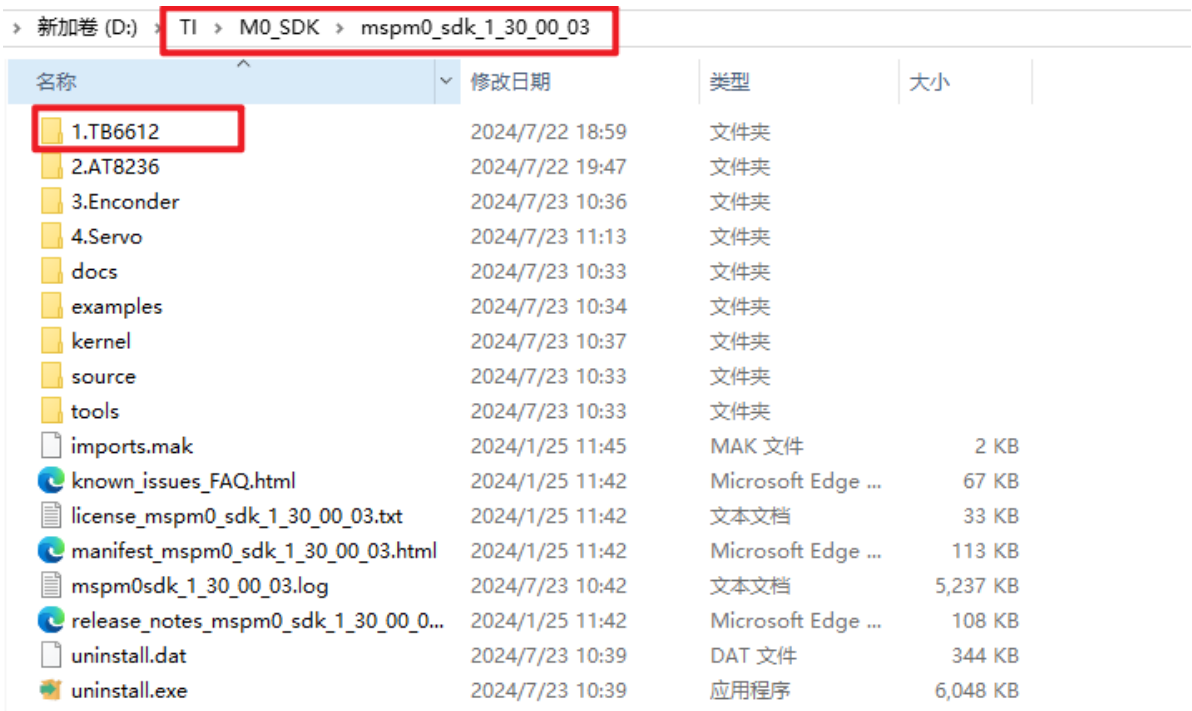
```
int main(void)
{
    //Development board initialization
    board_init();

    MPU6050_Init();

    uint8_t ret = 1;

    float pitch=0,roll=0,yaw=0;   //Euler Angles

    printf("start\r\n");

    //DMP Initialization
    while( mpu_dmp_init() )
    {
        printf("dmp error\r\n");
        delay_ms(200);
    }

    printf("Initialization Data Succeed \r\n");

    while(1)
    {
        //Get Euler angles
        if( mpu_dmp_get_data(&pitch,&roll,&yaw) == 0 )
        {
            printf("\r\npitch =%d\r\n", (int)pitch);
            printf("\r\nroll =%d\r\n", (int)roll);
            printf("\r\nyaw =%d\r\n", (int)yaw);
        }
        delay_ms(200);//According to the set sampling rate, the delay cannot be
set too large
    }

}
```

The main purpose of this code change is to initialize the MPU6050 and obtain the Euler angles (pitch, roll, and yaw) through the DMP (digital motion processor).

First, initialize the development board, then initialize the MPU6050. Next, we try to initialize the DMP through the mpu_dmp_init function. If the initialization fails, the error message is printed in a loop and delayed for a while. Once the DMP is initialized successfully, we enter an infinite loop, obtain the current Euler angles through the mpu_dmp_get_data function, and print them through the serial port.

Note: The project source code must be placed in the SDK path for compilation,

For example, path: D:\TI\M0_SDK\mspm0_sdk_1_30_00_03\TB6612


新加卷 (D:) › TI › M0_SDK › mspm0_sdk_1_30_00_03
名称
修改日期
类型
大小
1.TB6612 2024/7/22 18:59 文件夹
2.AT8236 2024/7/22 19:47 文件夹
3.Enconder 2024/7/23 10:36 文件夹
4.Servo 2024/7/23 11:13 文件夹
docs 2024/7/23 10:33 文件夹
examples 2024/7/23 10:34 文件夹
kernel 2024/7/23 10:37 文件夹
source 2024/7/23 10:33 文件夹
tools 2024/7/23 10:33 文件夹
imports.mak 2024/1/25 11:45 MAK 文件 2 KB
known_issues_FAQ.html 2024/1/25 11:42 Microsoft Edge ... 67 KB
license_mspm0_sdk_1_30_00_03.txt 2024/1/25 11:42 文本文档 33 KB
manifest_mspm0_sdk_1_30_00_03.html 2024/1/25 11:42 Microsoft Edge ... 113 KB
mspm0sdk_1_30_00_03.log 2024/7/23 10:42 文本文档 5,237 KB
release_notes_mspm0_sdk_1_30_00_0... 2024/1/25 11:42 Microsoft Edge ... 108 KB
uninstall.dat 2024/7/23 10:39 DAT 文件 344 KB
uninstall.exe 2024/7/23 10:39 应用程序 6,048 KB